Reset-Password Documentation

The Reset Password component provides a secure and user-friendly interface for creating and confirming passwords. It supports real-time validation, configurable password rules, strength indicators, and optional confirm password fields.

The component ensures better UX by giving immediate feedback on password requirements and match status, making it ideal for signup, password reset, or account update forms.

How to use

Basic Usage

The Reset Password component can be easily integrated into forms to provide secure password input with real-time validation, strength indicators, and optional confirm password fields. Below are some common usage scenarios:

```
<aava-reset-password
  label="Password"
  placeholder="password"
  [rules]="passwordRules"
  [showConfirmPassword]="false"
  [showPasswordStrength]="true"
  size="md"
  (passwordChange)="onPasswordChange($event)"
>
</aava-reset-password>


---

  onPasswordChange(state: PasswordValidationState): void {
    console.log('Password State:', state);
  }

  passwordRules: PasswordRule[] = [
    {
      message: 'At least 8 characters',
      validator: (pwd) => pwd.length >= 8,
    },
    {
      message: 'One uppercase letter (A-Z)',
      validator: (pwd) => /[A-Z]/.test(pwd),
    },
    {
      message: 'One lowercase letter (a-z)',
      validator: (pwd) => /[a-z]/.test(pwd),
    },
    {
      message: 'One number (0-9)',
      validator: (pwd) => /\d/.test(pwd),
    },
  ];
```

- Basic password input with default labels and placeholder.
- Real-time validation of password rules if configured.
- Optional password strength indicator can be enabled.

Size Variants

```
<div>
  <label>Extra Small</label>
  <aava-reset-password
    label="Password"
    placeholder="Small size"
    [rules]="passwordRules.slice(0, 2)"
    size="xs"
  >
  </aava-reset-password>
</div>

<div>
  <label>Small</label>
  <aava-reset-password
    label="Password"
    placeholder="Large size"
    [rules]="passwordRules.slice(0, 2)"
    size="sm"
  >
  </aava-reset-password>
</div>

<div>
  <label>Medium</label>
  <aava-reset-password
    label="Password"
    placeholder="Large size"
    [rules]="passwordRules.slice(0, 2)"
    size="md"
  >
  </aava-reset-password>
</div>

<div>
  <label>Large</label>
  <aava-reset-password
    label="Password"
    placeholder="Large size"
    [rules]="passwordRules.slice(0, 2)"
    size="lg"
  >
  </aava-reset-password>
</div>

<div>
  <label>Extra Large</label>
  <aava-reset-password
    label="Password"
    placeholder="Large size"
    [rules]="passwordRules.slice(0, 2)"
    size="xl"
  >
  </aava-reset-password>
</div>


---

  passwordRules: PasswordRule[] = [
    {
```

```
      message: 'At least 8 characters',
      validator: (pwd) => pwd.length >= 8,
    },
    {
      message: 'One uppercase letter (A-Z)',
      validator: (pwd) => /[A-Z]/.test(pwd),
    },
    {
      message: 'One lowercase letter (a-z)',
      validator: (pwd) => /[a-z]/.test(pwd),
    },
    {
      message: 'One number (0-9)',
      validator: (pwd) => /\d/.test(pwd),
    },
  ];
```

Available Sizes

The Reset Password component supports five size variants to fit different form layouts. The size affects the input field height, font size, and icon dimensions. Use smaller sizes for compact forms and larger sizes for more prominent input fields.

• xs (Extra Small) – Very compact input for dense or mobile-focused layouts
• sm (Small) – Slightly larger input for standard compact forms
• md (Medium) – Default size for most use cases (balanced proportions)
• lg (Large) – Larger input for prominent forms or accessibility-focused layouts
• xl (Extra Large) – Extra-large input for full-width forms or highly prominent password fields

Custom Rules & Styling

```
<aava-reset-password
  label="Corporate Password"
  placeholder="Enter your corporate password"
  [rules]="customPasswordRules"
  [showConfirmPassword]="true"
  [showPasswordStrength]="false"
  size="md"
  (passwordChange)="onPasswordChange($event)"
>
</aava-reset-password>


---

  onPasswordChange(state: PasswordValidationState): void {
    console.log('Password State:', state);
  }

  customPasswordRules: PasswordRule[] = [
    {
      message: 'Between 10-20 characters',
      validator: (pwd) => pwd.length >= 10 && pwd.length <= 20,
    },
    {
      message: 'No spaces allowed',
      validator: (pwd) => !/\s/.test(pwd),
    },
    {
      message: 'Must contain "Corp" or "Dept"',
      validator: (pwd) => /Corp|Dept/i.test(pwd),
    },
  ];
```

• Customize password rules with messages and validators.
• Apply custom CSS styles to match the form’s design system.

With Confirm Password

```
<aava-reset-password
  label="New Password"
  confirmLabel="Confirm New Password"
  placeholder="Enter strong password"
  confirmPlaceholder="Re-enter password"
  [rules]="passwordRules"
  [showConfirmPassword]="true"
  [showPasswordStrength]="true"
  size="md"
  (passwordChange)="onPasswordChange($event)"
  (validationChange)="onValidationChange($event)"
>
</aava-reset-password>


---

  onPasswordChange(state: PasswordValidationState): void {
    console.log('Password State:', state);
  }

  onValidationChange(isValid: boolean): void {
    console.log('Password is valid:', isValid);
  }

  passwordRules: PasswordRule[] = [
    {
      message: 'At least 12 characters',
      validator: (pwd) => pwd.length >= 12,
    },
    {
      message: 'One uppercase letter (A-Z)',
      validator: (pwd) => /[A-Z]/.test(pwd),
    },
    {
      message: 'One lowercase letter (a-z)',
      validator: (pwd) => /[a-z]/.test(pwd),
    },
    {
      message: 'One number (0-9)',
      validator: (pwd) => /\d/.test(pwd),
    },
    {
      message: 'One special character (!@#$%^&*)',
      validator: (pwd) => /[!@#$%^&*(),.?":{}|<>]/.test(pwd),
    },
    {
      message: 'No common words (password, 123456)',
      validator: (pwd) => {
        const commonWords = ['password', '123456', 'qwerty', 'admin'];
        return !commonWords.some((word) => pwd.toLowerCase().includes(word));
      },
    },
  ];
```

- Include a confirm password field to ensure user input matches.
- Provides immediate visual feedback when passwords do not match.

Accessibility

The Reset Password component follows WAI-ARIA and general accessibility best practices:

• All input fields are keyboard-navigable (Tab and Shift+Tab support).
• Password visibility toggle buttons are accessible via keyboard (Enter/Space) and have meaningful aria-label s.
• Validation messages (password rules, password match) are visible and screen reader-friendly.
• Proper focus management when displaying error or validation states.
• Uses semantic input elements ( ) for compatibility with assistive technologies.
• Visual indicators (strength bar, rule checks) are paired with text for screen reader users.

API Reference

Input Properties

Property	Type	Default	Description
label	string	'New Password'	Label for the password input field.
confirmLabel	string	'Confirm Password'	Label for the confirm password input f
placeholder	string	'Enter your password'	Placeholder text for the password fiel
confirmPlaceholder	string	'Re-enter your password'	Placeholder text for the confirm passv
rules	PasswordRule[]	[]	Array of password validation rules, ea
showConfirmPassword	boolean	false	Whether to show the confirm passwor
size	'xs' | 'sm' | 'md' | 'lg' | 'xl'	'md'	Size variant for the input fields.
disabled	boolean	false	Disables the input fields when set to t
showPasswordStrength	boolean	false	Whether to show a password strength

Output Events

Event	Type	Description
passwordChange	EventEmitter<PasswordValidationState>	Emits the current password state, including pas
validationChange	EventEmitter<boolean>	Emits true if the password (and confirm passwo

Best Practices

Content & Design

• Clearly label password and confirm password fields for clarity.
• Use concise, descriptive messages for password rules to guide users effectively.
• Display only relevant password rules to avoid overwhelming the user.

• Keep input fields visually distinct and properly sized based on the size input.
• Consider using icons (e.g., eye/eye-off) consistently for toggling password visibility.

Accessibility

• Ensure all password fields are keyboard-navigable.
• Provide meaningful aria-label or aria-describedby for assistive technologies.
• Use proper focus management when showing validation messages.
• Display clear error states for invalid inputs, including password mismatch or rule violations.
• Pair visual indicators (like strength bars) with textual cues for screen reader users.

Security & UX

• Avoid pre-filling password fields to reduce security risks.
• Validate passwords in real-time but avoid leaking rule-specific feedback that could aid attackers.
• Ensure password strength indicators are informative but not misleading.
• Use confirm password field only if necessary; hide it for simpler flows when appropriate.
• Provide immediate feedback on password match and rule validation for better user experience.

Performance

• Component uses Angular’s OnPush change detection for efficiency.
• Minimal event handling and lightweight DOM structure ensure smooth performance.
• Safe to use multiple instances on a single page.
• Real-time validation does not block rendering or degrade UI responsiveness.